CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Visualisation of Test Failure Data to Support Fault Localisation in Distributed Embedded Systems within the Automotive Industry

Bachelor of Science Thesis in Software Engineering and Management

MICHAEL JONES
RAFAEL DA SILVA MARTINS

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Visualisation of Test Failure Data Using a Similarity Based Technique:
Supporting Fault Localisation in Distributed Embedded Systems within the Automotive Industry

MICHAEL JONES
RAFAEL DA SILVA MARTINS

Supervisor: FRANCISCO GOMES DE OLIVEIRA NETO
Examiner: JENNIFER HORKOFF

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover:
Volvo XC 60 – hardware and software tested on the same test environment used for this thesis work.

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Visualisation of Test Failure Data to Support Fault Localisation in Distributed Embedded Systems Within the Automotive Industry

Michael Jones
Department of Computer Science and Engineering
University of Gothenburg
gusjonemi@student.gu.se

Rafael da Silva Martins
Department of Computer Science and Engineering
University of Gothenburg
gusdasra@student.gu.se

***Abstract*—In this thesis we present the design, development and evaluation of a software tool with the purpose of assisting in the localisation of root causes of test case failures in distributed embedded systems, specifically vehicle systems controlled by a network of electronic control units (ECUs). Fault localisation is especially hard in such systems due to its distributed nature, and often organisations rely on the knowledge of in-house specialists for detecting and rectifying the underlying root cause of test case failures. The study took place in-situ at the Research and Development division of Volvo Car Corporation, a large automotive manufacturer. Researchers had access to a vast number of test execution logs from large-scale software integration testing under a continuous integration process. The main objectives of the research were to develop and evaluate a data visualisation tool to support root-cause identification of failures in order to foster a continuous feedback loop in the fault localisation process. Our contributions encourage the improvement of testing quality and supporting the development and adoption of test case writing guidelines and test failure debugging procedures. The research concludes that the use of data visualisation techniques can considerably boost the failure debugging procedures by presenting data in a clear and concise manner and making use of test harnesses to directly assist in reducing possible causes of failures. Additionally it encourages a systematic and continuous analysis of the current state of testing by aggregating, categorising and displaying large amounts of historical data in a concise manner that allows stakeholders to identify patterns and trends in test results.**

Keywords*—Fault localisation, distributed embedded systems, automotive systems, fault prediction support, continuous integration, integration testing, failure data visualisation.

I. Introduction

The presence and significance of software within automotive vehicles is becoming evermore essential. Nowadays an average car is equipped with more than 70 distributed microcontrollers - also known as electronic control units (ECUs). ECUs are interconnected using five or more different communication networks sharing real-time data from sensors and control signals [1]. With the emergence of advanced functionalities within modern vehicles such as adaptive cruise control and advanced emergency braking [1], the degree of intricacy will definitely increase in the upcoming years.

A major challenge that faces the automotive industry is that of the integration testing of the distributed software. Proficient vehicle performance can only be maintained by supervising the convoluted interactions between a vehicles hardware (e.g. sensors, batteries and motors) [2] and software components and the algorithms they run on the embedded processors [3].

There are numerous potential faults that could be present in an automotive system. These faults can vary from a hardware or software fault on an individual subsystem, to a communication fault between numerous subsystems. The nature of these faults can be dynamic; for example, a hardware fault on one subsystem can cause a software fault on another. Two faults can also be symptomatic of each other [4]. Software fault localisation (diagnosing the root cause) can be a highly complex and time consuming task as a result of the anomalous nature and volume interactions between different components [1]. These factors lead to *software fault localisation techniques* being very expensive for automotive companies.

There is a very clear need for techniques that can reduce the time required by programmers and engineers to identify root causes of faults. Moreover, such techniques can have a major positive impact on the cost and quality of software development and maintenance [5].

Debugging tools and test execution reports normally use results of only one test execution instead of using the information provided by many executions of the software [6]. Moreover, test execution reports are mostly presented in textual form, making them increasingly hard to interpret for larger software and test suites.

Therefore, this study proposes using data visualisation techniques and statistical analysis to assist in the localisation of the root causes of failures. A better visualisation of test case failure data, with selection and prioritisation of relevant information can decrease the amount of time specialists spend on looking through test logs. Additionally, visualisation of statistics regarding type of failures and how often they happen will foster a continuous feedback loop in the fault localisation process, encouraging the improvement of testing quality by pointing out test cases trends and relevance (failure/success rates), consequently supporting the development and adoption of test case writing guidelines and test failure debugging procedures.

Human fault localisation is reliant on the experience of the person. The task can be very expensive and time consuming due to the fact that the manifestation of the error (failure) and its internal cause (fault) may have no obvious relationship [7].

A. Definitions

In order to keep consistency and being aware that some companies may have specific terminology for distinct terms in software/systems development, we present the terms used in our Thesis and what they refer to in our context. The terms and corresponding definitions are:

- *Fault localisation* - The process to find the location of faults. It determines the root cause of the failure. It identifies the causes of abnormal behaviour of a faulty program. It identifies exactly where the bugs are.
- *Continuous Integration* - A development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.
- *Test Case* - A set of conditions or variables under which a tester will determine whether a system under test satisfies requirements or works correctly.
- *Test Suite* - A collection of test cases that are intended to be used to test a software program to show that it has some specified set of behaviours.
- *Test Harness* - A collection of software and test data configured to test a program unit by running it under varying conditions and monitoring its behaviour and outputs.
- *Build* - A version of a program. As a rule, a build is a pre-release version and as such is identified by a build number, rather than by a release number. Reiterative builds are an important part of the development process. Throughout development, application components are collected and repeatedly compiled for testing purposes, to ensure a reliable final product.
- *Theme* - A theme captures something important about the data in relation to the research question and represents some level of patterned response or meaning within the data set.
- *Thematic Coding* - Thematic coding is a form of qualitative analysis which involves recording or identifying passages of text or images that are linked by a common theme or idea allowing you to index the text into categories and therefore establish a framework of thematic ideas about it [8].

B. Research Questions

- Main question:
  - RQ 1: How can we leverage fault localisation procedures in automotive system testing?
- Sub questions:
  - RQ 1.1: What information from automated integration testing is valuable in the fault localisation process?
  - RQ 1.2: How can we use test harnesses to support continuous feedback of testing activities?
  - RQ 1.3: How consistent are test case failures throughout extensive periods of integration tests?
- Assumptions:
  - Access to an automated test execution environment
  - Test execution environments (test rigs) up-and-running
  - In-house testing specialists will provide enough time to collaborate with the research

C. Structure of the Thesis

Section II discusses related work to our research and highlights how this research contrasts. Section III will outline the environment that the research was conducted in and the available resources. The motivation of the research and the proposed solution is explained in this section as well as the scientific and technical contributions that the research provides. Section IV details the methodologies that were adopted to conduct the data collection, and how the data would be utilised and presented. Finally, the methodology for the evaluation of our work is presented. Section V presents the artefacts that were produced and details their functionality. Section VI firstly analyses the evaluation of the artefacts by interpreting the findings of the interviews and then provides graphs that contribute to answering the research questions. The threats to validity to the undertaken research are discussed in section VII. Section VIII concludes the research by addressing the research questions that were proposed. Finally, section IX discusses potential continuations of our research and what more could have been done during this research.

II. Related Work

A. Relevant literature from the problem domain

There are a few references regarding fault localisation in distributed embedded systems. Kodali et al. [1] introduces an initial study on fault diagnosis for the Electric Power Generation and Storage System in a car, where they developed a multilevel scheme to diagnose hardware, software and their interactions. They validated their approach using an automotive simulation; on the other hand our research was evaluated in a testing and production environment for vehicle level systems.

Chunduri [9] introduces a verification strategy for testing distributed automotive embedded software functions and proposes a promising approach focused on identifying test gaps and redundancies. However they do not address identification or prediction for root causes of test failures.

Regarding the identification of failure causes, Mariani and Pastore [10] present a technique to analyze log files and retrieve relevant information to identify the causes. Their work on parsing, classifying and analyzing system logs for automated identification of failure causes is very relevant to our work, but it is not done within the higher complex context of distributed embedded systems in the automotive industry.

Relevant studies also include Seth and Khare [11] analysis on automated continuous integration using Jenkins and

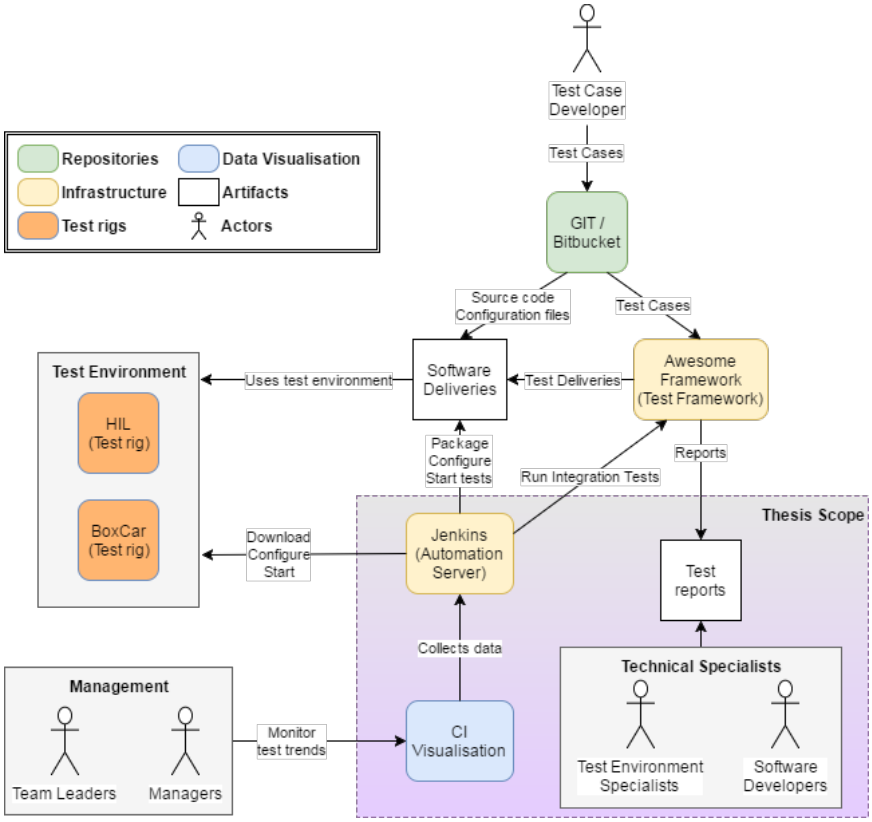


Fig. 1: High Level View of the Problem Domain

Pattipati et al. [12], who proposes an integrated diagnostic process composed of 6 steps: model, sense, develop and update test procedures, infer adaptive learning and predict. Similar to Kodali et al. [1], the researchers do not apply their proposals to a real production environment.

B. Literature on potential solution approaches

Jones et al. [6] proposes a new technique that also uses visualisation to assist in fault localisation. They propose the use of colour to visually map the participation of each program statement in the outcome of a program execution. Albeit their solution is beneficial towards fault localisation, their scope is much narrower as it focuses only on executed lines of code. Therefore, the use of their approach would be limited within a test execution where failures may emerge not only from software issues, but also from the test environment, the hardware and the communication links between the parts.

As far as the researchers are concerned, there is no published literature on data visualisation of test execution data. Telea's book [13] is a valuable source of information regarding the general principle and practices of data visualisation, but it is not concerned with the specifics of software test execution data.

The problem of diagnosing the root cause of a failure has been addressed by most of the researchers cited in the previous section. Kodali et al., Chunduri and Pattipati et al. [1, 9, 12] discuss possible approaches and models to solve the issue - mostly focusing on modelling. Mariani and Pastore [10] do not focus on distributed embedded systems, but rather on comparing successful test cases logs with failed ones and inferring models from the discrepancies. Due to the limited amount of information logging in VCCs test cases, we were unable to reproduce Mariani and Pastores [10] approach. Therefore, we focused on categorising the failure data by type and test case using a similarity based approach.

III. Research Context and Scope

The research takes place in-situ at the Research and Development division of Volvo Car Corporation (VCC), a large automotive manufacturer. The continuous integration team at VCC is responsible for improving the automated testing process of distributed embedded systems, specifically vehicle systems controlled by a network of electronic control units (ECUs). Jenkins [1] is being used as the integration server and the test automation framework that is in place is AwesomeFramework (in-house developed framework for test automation, based on the Robot Framework [2]). Researchers had access to a

[1]https://jenkins.io - Jenkins is a continuous integration software tool for testing and reporting on isolated changes in a larger code base in real time.

[2]Robot Framework - Generic Test Automation Framework for Acceptance Testing, http://robotframework.org/

significant amount of test execution logs from large-scale software integration testing under a continuous integration process. The current testing consists of over 30 test suites containing a total of over 250 test cases.

An overview of the current fault localisation process is shown in Figure 1. The automation server (Jenkins) manages the execution of test suites on new software deliveries. Jenkins run those tests using the test framework on different test environments. Test suite results are reported by the test framework, while a high level summary of the test cases successes/failures is displayed by the CI visualisation system. Technical specialists use the test reports in order to identify failed test cases and start the debugging processes, while the management team monitors the overall statistics via the CI visualisation system.

A. Problem Statement and Proposed Solution

The objective of the thesis is to investigate and implement a test failure data visualisation tool to support the process of fault localisation. Software fault localisation can be a highly complex and time consuming task as a result of the anomalous and distributive nature of the system, combined with the large volume of interactions between different components. These factors lead to software fault localisation being very expensive for automotive companies [1].

Currently there are no guidelines or data visualisation support for fault localisation at VCC. The organisation currently relies on in-house specialists who can diagnose the root cause of failures based on their knowledge and experience of the complex system. The current visualisation of test results is a test report from Awesome Framework containing an enormous amount of data regarding all test cases including those that did not fail. There is no significant work on improving the visualisation of these results, and finding relevant information regarding a test case failure can be a lengthy process.

A better visualisation of test case failure data, with selection and prioritization of relevant information can decrease the amount of time specialists spend on looking through test logs. Additionally, visualisation of statistics regarding type of failures and how often they happen will foster a continuous feedback loop in the fault localisation process, encouraging the improvement of testing quality by pointing out test cases trends and relevance (failure/success rates), consequently supporting the development and adoption of test case writing guidelines and test failure debugging procedures.

In summary, our contributions are:

1) *Scientific:*
   a) Classification tree of test case failures
   b) Systematic test log analysis
2) *Technical:*
   a) Test case database
   b) Scripts for automatically retrieving data and updating the database
   c) The visualisation platform

IV. Methodology

The thesis adopted a design science methodology for the research project (Figure 2). The tool that was designed and developed is a technology-based artefact that is a human and computer interface. Because of the intention to improve the performance of the artefact based on its relevant business problems, this methodology was optimal.

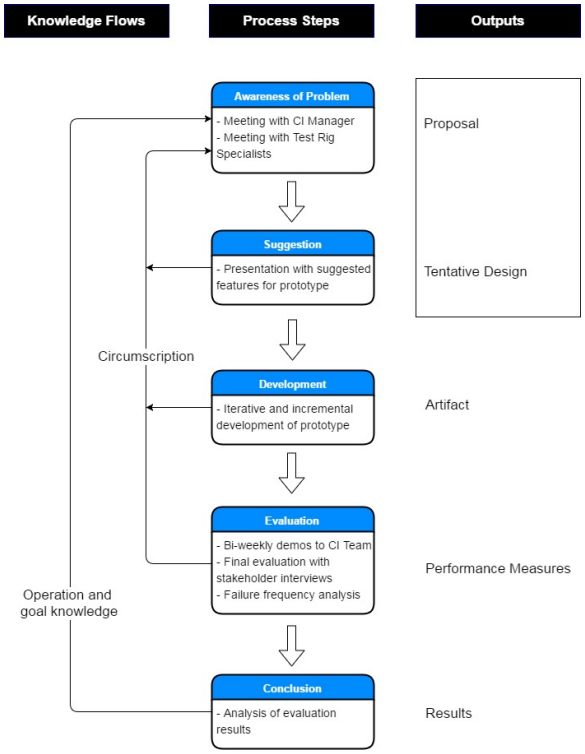


Fig. 2: Design science research methodology used in this work.

The prototype was designed and developed in an iterative manner and demonstrated to stakeholders at regular intervals. The artefact was evaluated rigorously and presented from both a technology and management perspective. The use of graphs and charts was effective in rendering the results of our artefact to stakeholders of all backgrounds. The research conformed to the guidelines of design sciences by providing clear research contributions mentioned earlier, and communicating the research to both technology-orientated and management-orientated audiences.

The data visualisation tool consists of four modules: test data collection, test data storage, querying of stored data and visualisation. The tool is being implemented as part of the existing CI chain.

A. Test Data Collection and Storage

The main source of information for this study are test execution logs from integration testing within a Continuous Integration system. Researchers had access to an extensive

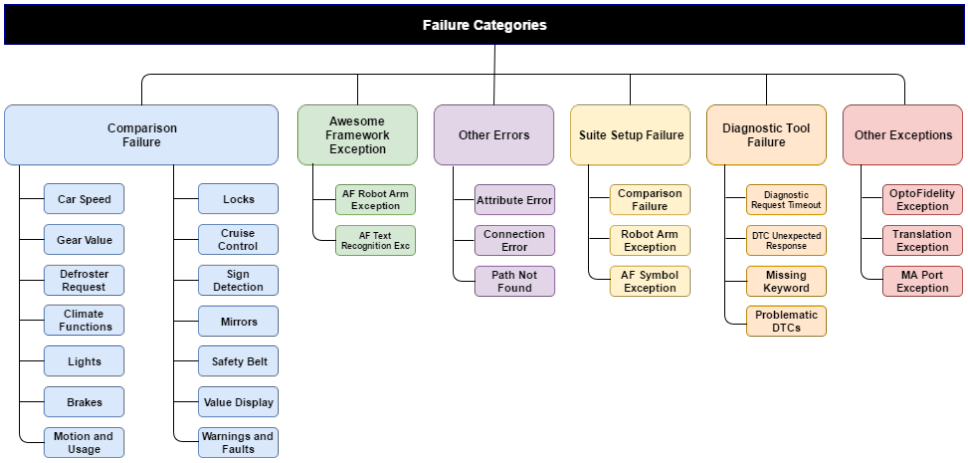


Fig. 3: Categorisation of types of failures.

dataset of past and present test results consisting of over 11,500 individual test case executions logs. The data represents tests executed in one test rig over the last 6 months. Additionally, researchers were able to trigger and collect new data from testing when necessary. Specific data includes a list of passed and failed test cases, console logs from tests and which software deliveries triggered testing. Additionally, researchers had access to specialists who have in-depth knowledge of possible root-causes for specific test failures.

The current testing framework that is in use by Volvo Cars Corporation is derived from the open-source Robot framework. The framework outputs test execution logs that were the primary source of data collection. The logs are generated in both HyperText Markup Language (HTML) and Extensible Markup Language (XML) format. The HTML format is more appropriate for human use because of its readability. XML is the format that is ideal for machine reading - because of this the scripts written were tailored for the XML format. Such scripts were responsible for automatically fetching, processing and storing the data.

Initially, a python [3] script was written to fetch and store past test results generated by the automated testing framework AwesomeFramework. The XML tree was parsed for every executed test case and data was extracted, categorised and stored in a local SQLite [4] database. The database has been designed, developed and implemented by the researchers.

B. Failure Categorisation

A failure classification tree was developed in order to improve the visualisation of different test areas. The initial tree design was created after careful analysis of all test case failures. The failure type was used as the main classification criteria, and was later subdivided regarding to different criteria, such as the functional area it affected or the ECU it was related to.

After the failure categories were defined, they were used to classify collected test case failure data. Test data parsing and storing scripts were updated to include this information. Failure categories were later validated with stakeholders during the interview evaluation of the prototype.

C. Data visualisation

A prototype of the failure data visualisation tool has been developed using an iterative-incremental approach. Key factors were taken into consideration when developing the visualisation template:

- Initial focus on the last test build
- Clear presentation of the project and build number
- A quick summary of the test results using graphs based on the failure categories
- A list containing information about failed test cases which is considerably easier to read than the current test logs
- Information on *new failures* (failed test cases that have not failed before)
- Information on *frequent failures* in the form of a table that allows filtering by number of builds and percentage of failed runs.

D. Evaluation

A qualitative approach was adopted for the evaluation of the visualisation tool. The purpose of the evaluation was to assess how the produced artefact compares with the existing visualisation; in order to do this it was necessary to have participants familiar with the existing VCC environment.

Essentially a qualitative approach helps researchers to understand a phenomenon from the point of view of the participants and its particular social and institutional context [14]. This approach aided the researchers in understanding and interpreting the perceptions that the participants have of the visualisation

[3]Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org

[4]Hipp, D. R., Kennedy, D., Mistachkin, J., SQLite, version 3.8. Available at https://www.sqlite.org

tool. This methodology was selected due to the precise context of the research; relevant to VCC and its testing environment. The limited number of relevant potential participants available also restricted the potential use of statistical analysis to draw any meaningful conclusions from the data that was collected.

By conducting semi-structured interviews with open ended and follow up questions based on the topics that needed to be covered, the researchers were enabled to probe beyond the initial responses [15]. A list of questions was created (see Appendix A) that would be asked, however, throughout the interview process the researchers were granted freedom to ask the interviewee impromptu questions based on their responses and build a synergy between the interviewees and the researchers.

The questions that were drawn up were done in an unbiased manner which reduced the risk of influencing the participants responses. The interviews were recorded to allow interpretation to be undertaken after the interviewing process has finished. The interviewees were selected to provide feedback from multiple stakeholder groups. The following stakeholders were selected:

- Test rig analysis team (2 members, 1 joint interview)
- Continuous integration team leader (1 member, 1 interview)

These stakeholders provided both managerial and developing outlooks. The team leader provided the view point of how the testing environment was performing, whilst the test rig analyser showed more concerned with the aspects relevant to each build and the fault localisation process.

The interviews were transcribed for the purpose of analysing their feedback. Analysis of the data was done with the use of thematic analysis. This is a method for identifying, analysing and reporting patterns within data. [16]. It emphasises organisation and rich description of the data set by identifying both implicit and explicit ideas within data [8]. Themes from the data are developed primarily through the use of coding; this is the process of recognising important moments in the data [17]. This method facilitated the researchers in interpreting the participants feedback on the new visualisation techniques. Results from the thematic analysis were presented and discussed with the stakeholders at the end of the study.

V. Results

A. Failure Categorisation Tree

The classification tree was constructed by analysing the database of test case failures from every build to date. The warn and error messages that were output by the test cases were investigated rigorously and categories and sub-categories were established according to the dissimilarity of messages. The failure classification tree created by the researchers is visualised in Figure 3. Failures were classified using three levels:

- Level 1 - Failure Category: differentiates types of failures by the main type of test execution failure. Examples include Comparison Failure (when an assertion test for a specific value fails), Diagnostic Tool Failure (when the diagnostic software used reports a failure) and other software execution specific failures.
- Level 2 - Failure Breakdown: categorises the types of failure using common denominators between them. For example, *Comparison Failures* are further categorised in regard to the part of the system they belong to, while *Diagnostic Tool Failures* are broken into the specific ECU that signaled an error message and other communication errors.
- Level 3 - Specific Failure: a more refined level of classification, currently not in use by the visualisation system. An example would be which type of gear was tested (automatic or manual), or which defroster signal has raised an error (rear or side mirror defroster).

B. Failure visualisation Tool

The visualisation takes the form of web pages that are automatically generated by a script running in Jenkins. The script was created and added to the test execution runs by the researchers. A current prototype can be seen in Figures 4 and 5. The pages consist of interactive graphs and tables that highlight the most relevant data needed to determine the potential root cause(s) of test case failures.

A combination of test execution data, the failure classification tree and statistics on previous builds is used to generate the failure visualisation. Figure 4 displays the following data:

- Project name and build number on the top of the page
- A bar chart representing the count of each Failure Category that was present in the build.
- A pie chart to portray the breakdown of a selected Failure Category by displaying counts of its sub-categories.
- A table of every test case failure that will display: *i.)* Test suite name *ii.)* Test case ID *iii.)* Test case name *iv.)* Failure Category *v.)* Warn message *vi.)* Fail message.

The bar chart offers interaction to the user so that when a column is clicked, the pie chart and table of test case failures will update with the data relevant to that category. The tool will also provide summaries of how the testing process is performing on a longer term basis. The following information is also presented to the user (Figure 5):

- A table for test cases that have never failed.
- An interactive table of test cases that have been frequently failing over the latest number of specified builds. This includes a filter for the failure frequency (50%, 75% or 100% occurrence).

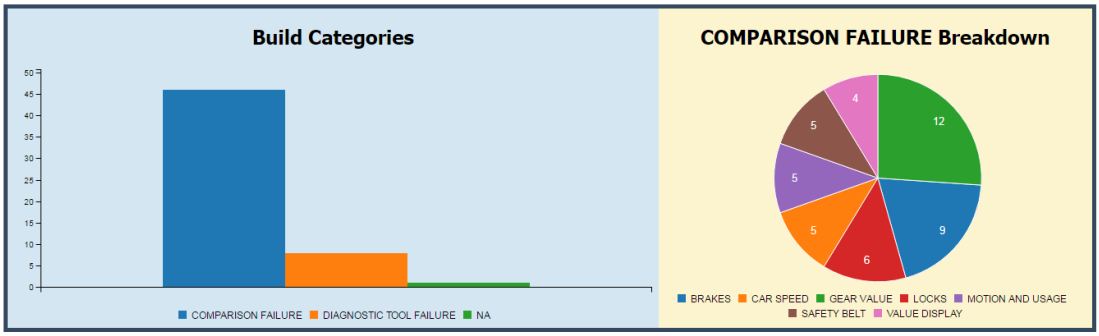
A long term perspective of test result data in the context of distributed embedded systems should highlight to both developers and other stakeholders what area of the testing rig needs to be improved to facilitate more stable testing environments.

C. Failure Frequency Analysis

Data analysis has been performed in order to demonstrate the potential of the proposed failure data collection and visualisation system in improving different aspects of test quality. The examples in this section are not yet implemented in the data

NightTest HIL - Build 104

Failures Summary



Test Case Failures

Test Suite	Test Case ID	Test Case Name	Category	Warn Message	Fail Message
HIL » HIL stability 94032 » BasicDrivingCycle	s1-s2-t5	Simple drive cycle - slow acceleration - soft braking	COMPARISON FAILURE	FAILED CHECK: car speed is greater than or equal to [10]=10.0 -- Actual value: 2.53368 [m/s]	Comparison failure
PKE » PKE Verified » PKE 017 S-17-04 GearPositionWarning	s1-s4-t1	S-17-04 Gear Position Warning	COMPARISON FAILURE	FAILED CHECK: usage mode is driving -- Actual value: UsgModInActv	Comparison failure
PKE » PKE Verified » PKE 017 S-17-04 GearPositionWarning	s1-s4-t2	Disengage EPB and drive off	COMPARISON FAILURE	FAILED CHECK: electrical parking brake status is released -- Actual value: applied	Comparison failure
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t1	S-22-04 L1-Ratio Management	COMPARISON FAILURE	FAILED CHECK: usage mode is driving -- Actual value: UsgModInActv	Comparison failure
PKE » PKE Verified » PKE 022 S-22-04 L1-	s1-s5-t3	Verify gears changing	COMPARISON	FAILED CHECK: actual gear is 1 --	Comparison failure

Fig. 4: Failure data visualisation page, top of the page.

New Failures

Frequent Test Case Failures

Select number of latest builds 5

Select Failure Frequency 100%

Test Suite	Test Case ID	Test Case Name
HIL » HIL stability 94032 » BasicDrivingCycle	s1-s2-t5	Simple drive cycle - slow acceleration - soft braking
HIL » HIL stability 94032 » BasicDrivingCycle	s1-s30-t1	Start car from Usage Mode Inactive
HIL » HIL stability 94032 » BasicDrivingCycle	s1-s30-t3	Cycle through different Usage Modes
HIL » HIL stability 94032 » DtcReadoutsSpa	s1-s29-t11	DTC Test IHU
HIL » HIL stability 94032 » DtcReadoutsSpa	s1-s29-t13	DTC Test PDM
HIL » HIL stability 94032 » DtcReadoutsSpa	s1-s29-t3	DTC Test BCM
HIL » HIL stability 94032 » DtcReadoutsSpa	s1-s29-t4	DTC Test CCM
HIL » HIL stability 94032 » DtcReadoutsSpa	s1-s29-t5	DTC Test CEM
HIL » HIL stability 94032 » DtcReadoutsSpa	s1-s29-t6	DTC Test DDM
HIL » HIL stability 94032 » DtcReadoutsSpa	s1-s29-t7	DTC Test DIM
PKE » PKE Verified » PKE 017 S-17-04 GearPositionWarning	s1-s4-t1	S-17-04 Gear Position Warning
PKE » PKE Verified » PKE 017 S-17-04 GearPositionWarning	s1-s4-t2	Disengage EPB and drive off
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t1	S-22-04 L1-Ratio Management
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t3	Verify gears changing accordingly with 60% throttle
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t4	Verify vehicle reaching 6th gear with 40% throttle
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t5	Verify vehicle reaching 7th gear with 40% throttle
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t6	Stop vehicle and put gear in R
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t7	Drive off in R
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t8	Verify correct gear when rolling backwards
PKE » PKE Verified » PKE 022 S-22-04 L1-RatioManagement	s1-s5-t9	Stop vehicle and put gear in P
PKE » PKE Verified » PKE 066 M-66 SafetyBeltReminder	s1-s10-t1	M-66 Safety Belt Reminder

Fig. 5: Failure data visualisation page, new and frequent failures section.

visualisation tool. One of the possible areas of improvement is analysing the failure frequency of test case failures through time. As an example, Figure 6 shows the average number of test case failures in 10-build clusters for each test suite present in the collection of Hardware-in-loop (HIL) Test Suites.

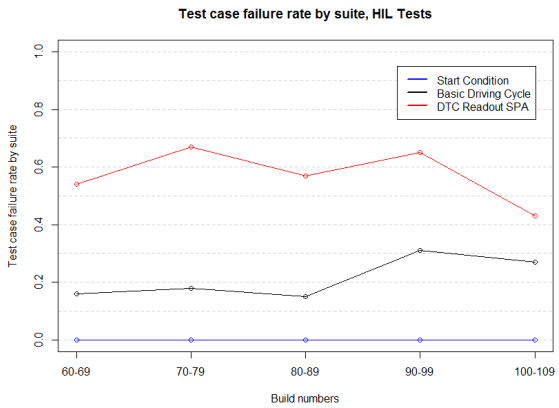


Fig. 6: Ratio of failed test cases per test suite for every 10 builds, HIL Tests

Figure 7 presents an example of visualisation of test case results for a range of build executions. The test suite presented runs a set of 7 test cases, executing the test set twice. Green cells denote passed test cases, yellow cells represent test cases that were not run and red cells display test case failures.

VI. Analysis

A. Interview Feedback

The first stage of thematic analysis involves the familiarisation of the researchers with the collected data. The interview evaluation was used to answer research questions 1, 1.1 and 1.2. Both researchers were present in the interviewing process, and the following step was to parse the interview notes for codes. Appendices B and C contain the notes from the interviewers. Items in bold depict codes that were extracted from the text. The codes were then refined, table 1 shows the initial codes that were created from both data sets.

The codes that were most frequent and held significant interest to the research questions were formed into themes or sub-themes. After reviewing all codes and themes, some of the codes were discarded as they did not fit into any of the constructed themes or hold significance to the research questions.

Figure 8 illustrates the relationship of themes, sub-themes and the codes which they capture. The themes and sub-themes are defined as the following:

- Themes:
  - *Visualisation* - Refers to the web page that was created and all data that is presented in the page.
  - *Fault localisation* - Captures all aspects that can contribute to the process of locating faults within the automated testing.

Codes	
Test Rig Analysers	CI Team Leader
Previous Builds	Starting point
Filters	Expansion
Hyperlink to HTML log	More Information
Build Comparison	Investigation
Suite Setup Failure	Automation
Tags	Test Case Performance
AwesomeFramework	Archive of Test Case Information
Category definitions	Interaction
States	Machine learning
Test case performance	Never failed
Software / hardware versions	Fault localisation
More Data / Information	Statistical Analysis
New failures	Logging
DTC	
Trends	
Archive	

TABLE I: Initial codes from interview transcript

  - *Future Work* - Highlights the potential additional features that were identified in the interview.
- Sub-themes:
  - *Test Case Performance* - Encapsulates all aspects of how test cases are performing over one or more builds.
  - *Comparison with Previous Builds* - The use of comparing the latest build with previously executed builds with the purpose of highlighting similarities or differences.
  - *Statistical Analysis* - The application of statistical methods to analyse a build with the means of predicting or outlining data.

Figure 8 illustrates what the interviewees were concerned with in regards to the fault localisation process, and what features should be added in the future to contribute to this process. The most important aspects of the current visualisation were also identified - interactive filters for tables and charts to be able to select only the relevant data that the user is interested in. The significance of having a link to the HTML AwesomeFramework report containing more context and detail on the test case failure was also highlighted. The participants were concerned with the fact that the visualisation may remove too much information. The performance of test cases was a major aspect of the interviewees concerns which was identified through the use of thematic analysis.

The inclusion of categories for test case failures was discussed - the test rig analysis team indicated that the sub-categories could be more well defined, even pointing out a specific case involving the parking or electrical brake.

The managerial perspective showed interest in the never failed test cases whereas the test rig analyser were more concerned with the new failures feature. The central talking point throughout both interviews was the ability to be able to compare the latest build with previous builds. Although the

Test Case \ Build Number	62	63	64	65	66	67	68	69	70	74	76	77	78	79	80	81	82	83	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105	106
Start car from Usage Mode Inactive	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Check vehicle motion status	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cycle through different Usage Modes	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	1
Clear DTCs in all ECUs	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	-1	1	-1	1	1	1	1	1	1	1	1	1	-1	1
Simple drive cycle - slow acceleration - soft braking	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Simple drive cycle - fast acceleration - hard braking	1	1	1	1	1	-1	-1	-1	1	1	1	-1	1	1	1	-1	1	1	1	1	1	1	1	-1	1	1	-1	-1	1	1	1	1	1	-1	1	1	-1	1
Stop and turn off vehicle	1	1	1	1	1	1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	1
Start car from Usage Mode Inactive	1	1	1	1	1	1	1	-1	0	1	1	1	1	1	1	-1	1	1	1	1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	1
Check vehicle motion status	1	1	1	1	1	1	1	-1	0	1	1	1	1	1	1	-1	1	1	1	1	-1	1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cycle through different Usage Modes	1	1	1	1	1	1	1	-1	0	1	1	1	1	1	1	-1	1	1	1	1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	1
Clear DTCs in all ECUs	1	1	1	1	1	1	1	-1	0	1	1	1	1	1	1	-1	1	1	1	1	-1	1	-1	1	1	1	-1	1	1	1	1	1	1	1	1	1	-1	1
Simple drive cycle - slow acceleration - soft braking	1	-1	1	1	1	-1	-1	-1	0	1	1	1	1	1	1	-1	1	1	1	1	-1	1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	1
Simple drive cycle - fast acceleration - hard braking	1	-1	1	1	1	-1	-1	-1	0	1	1	1	1	1	1	-1	1	1	1	1	-1	1	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1	1
Stop and turn off vehicle	1	1	1	1	1	1	1	1	-1	0	1	1	1	1	1	1	-1	1	1	1	1	-1	1	-1	1	1	1	1	1	1	1	1	1	1	1	1	-1	1

Fig. 7: Visualisation of test case results from the Basic Driving Cycle test suite over several builds

current visualisation does this to an extent, the interviewees identified that there was scope for improvement. By adding functionality to compare software and hardware versions of previous builds, this may indicate potential causes of test case failures or what should be investigated.

The new failures feature was recognised as having a positive impact and the never failed feature was great for identifying issues that may need to be investigated.

The test rig analysis team proposed that visualising the different states of test case failures as represented in the classification failure tree could help the fault localisation process, by comparing states with previous test case failures.

Having an archive for each test case with information that can be added and deleted multiple users would be beneficial in the fault localisation process. This information could indicate to a user that a test case might be repeatedly failing for a specific reason and that it may not need to be investigated.

The inclusion of tags into the visualisation with filters would be beneficial to users. Having the features to remove irrelevant test case failures and only display the failures that they are interested in is a features which would save vast amounts of time compared with the existing visualisation at VCC.

The CI team leader indicated that statistical analysis of test data could play a major role in the fault localisation process. Along with the introduction of machine learning this would facilitate a self-adapting fault localisation tool that would maximise its efficiency through iterative learning. This interview indicated that the work here provided solid foundation for the continuous transition towards fully automated testing and fault localisation.

B. Failure Frequency Analysis

Figures 6 and 7 are visualisation proposals for test case frequency analyses. These results are used to address research question 1.3. Figure 6 is an example of how to present test case data using a grouping convention against build execution clusters containing data from a specified number of runs. In this example, the tests are grouped by suites and each build execution cluster contains data from 10 test runs. The test case groups and build clustering can be modified for different purposes. An idea would be to use the failure categories instead of test suites, and to add interaction for data manipulation. An example of interaction could be having a slider to change the number of builds per cluster.

The example shown in Figure 6 gives a good overview of how the test suites from the HIL Tests test package are performing. In this case, it is visible that no test case from the Start Condition suite has ever failed. Such result implies that the executed tests are not able to detect failures at all, which could imply that current tests are not effective in detecting faults. On the other hand, over 40% of test case in the Diagnostic trouble code (DTC) Readout Tests tend to always fail. In this case, test cases are detecting faults - but somehow those faults seem not to be addressed. With this information in hand, managers and specialists should be able to better direct their focus to solve these recurring issues.

Figure 7 provides an in-depth analysis of test case results from the Basic Driving Cycle test suite. This type of visualisation allows stakeholders to pinpoint issues with individual test cases. In this example, the test case named *Simple drive cycle - slow acceleration - soft braking* has failed in all but one execution build. Such results are a clear indication that something is wrong. Possible causes could be a badly written test case, or improper tear down of previous test case conditions.

There is a plethora of additional information that can be extracted from these two simple examples, which strengthens the claim for the use of data visualisation techniques in failure analysis.

VII. Threats to Validity

Empirical research methods are prone to a number of validity threats [18]. The primary types of validity for this work were identified and are discussed in this section.

A. Construct Validity

Construct validity refers to which degree the studied operational actually measures what it claims to be measuring [19]. The main construct validity to this study was regarding to the design of the interview questions that could lead to researchers collecting irrelevant data to what was being studied. This risk was mitigated by having a more experienced third party review and revise the questions.

B. Internal Validity

Internal validity deals with the extent to which a causal conclusion based on a study is warranted, which in turn is

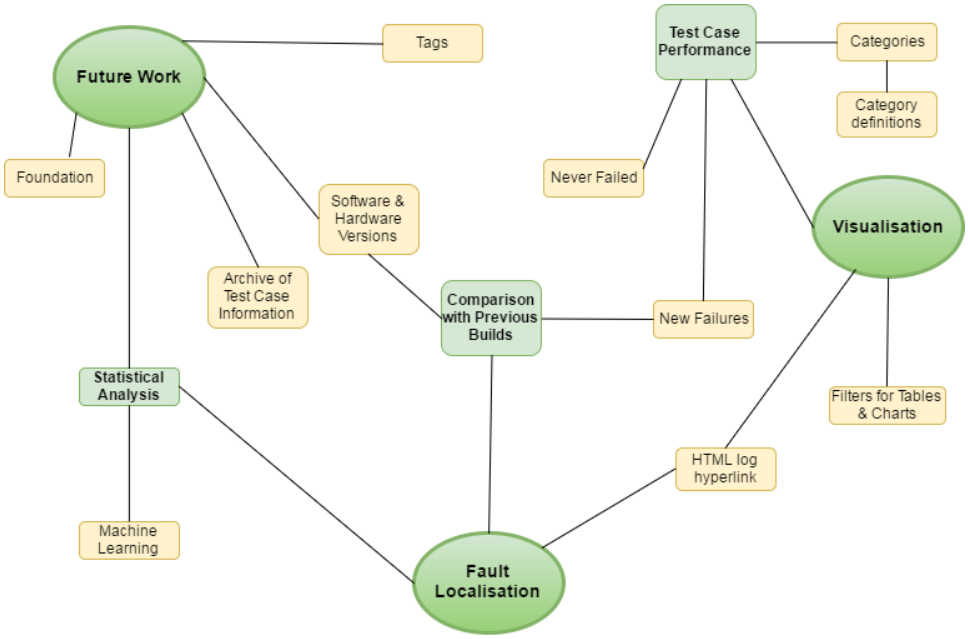


Fig. 8: Illustration of thematic analysis of interviews

determined by the degree to which a study minimises systematic error [19]. There was a risk of bias when analysing the interview transcripts. This risk was mitigated by performing the interview analysis based on scientific methods (thematic analysis) and having those results discussed and validated by the stakeholders in the study.

There is still a possible validity threat related to the limited number of stakeholders and interviews during the evaluation of the artefact. However, it was safer from the company's perspective to perform this initial study in a small scale. This threat can still be mitigated in a future work consisting of more robust evaluation methods on a more extensive time frame.

C. External Validity

External validity refers to the extent which the findings can be generalised to other environments [19]. Since the research was conducted at only one automotive company, results may fail to be reproducible in other companies. This risk was mitigated by having a very well defined context within which the study was conducted. Additionally, the characteristics of VCC's specific case were properly outlined in this work.

Additionally, we were systematic when designing the fault tree, but not general enough. This risk could be mitigated by having access to additional sets of test suites and test cases, which would help to determine more general patterns in the failure data.

Nonetheless, this is a stepping stone towards a continuous integration culture within Volvo, and it is safer from the companys perspective to begin with a small study with lower risks involving fewer people and limited scope within the company itself.

VIII. Conclusion

In this thesis we have proposed a test case failure data visualisation system that can support the fault localisation procedures. The system combines a failure classification tree and existing data from previous and current test builds to generate a test report that includes both the status of the latest test runs and archived information about previous runs. The failures are categorised and summarised in charts for a quick visualisation of the test results. Furthermore, a table containing a list of failed test cases is displayed. This table can be filtered by selecting the different categories on the charts. The information is kept at the minimum necessary, but more context is provided via hyperlinks to the complete failure logs. Additional information is shown regarding the long term status of the testing scope - a section containing a list of test cases that failed for the first time ever, a list of test cases that have never failed and an interactive table displaying test cases that have been failing often in the last n builds.

The thesis also presented the results of two interviews with main stakeholders that were used to evaluate the effect of the visualisation technique. The results have shown that the proposed data visualisation has a positive impact on the fault localisation process - and suggests promising directions for future work.

Answering our main research question, and according to our findings, Fault localisation procedures can be leveraged in the following ways:

- Categorising and displaying failures as charts allows the test rig specialists to quickly identify areas that require more immediate attention. Being able to filter failures by those areas helps them to focus on what needs to be addressed first.

- Minimising the amount of data regarding each test case and adding hyperlinks for additional context reduces the excess of superfluous information.
- Information that combines data from multiple previous runs in a single page saves the specialists' time and effort, as they do not need to crawl through multiple previous logs to access information regarding test cases results.

The interview also addressed the first research sub-question by identifying what type of information from the automated integration testing setup is valuable in the fault localisation process. Test rig specialists considered that data regarding new test case failures and the statistics including results from several test builds had a positive effect on the fault localisation process, as this information was not directly available before. The presence of these features has also encouraged them to suggest possible new additions to the visualisation page, such as a build comparison tool so that two similar runs can be directly compared and their differences easily highlighted. Such an addition would further improve their ability to pinpoint faults.

Our second research sub-question was related to the use pre-existing test harnesses to support continuous feedback in the fault localisation process. When presented with our artefact, all interviewees felt compelled to analyse the current fault localisation process and encouraged to initiate discussions on existing problems and how to use existing harnesses to provide possible solutions.

Finally, the third research sub-question is answered by our showcase of the possibilities to use such a system to monitor test case failure consistency over extensive periods of integration testing. This is only one in a myriad of possible applications of statistical data analysis and visualisation uses to boost the fault localisation procedures.

IX. Future Work

The research from this project outlined which data is significant in the fault localisation process. The extra features outlined in the interviewing process could be implemented and then the tool could be re-evaluated to measure it's new effectiveness. The impact of statistical analysis and machine learning on the fault localisation process is also something that could prove to be very beneficial. Research undertaken here provides the foundation for this work to commence, the database that has been constructed is the necessary platform and concept for the statistical approach to function on. One of the difficulties faced to the researchers was the number of participants available for the evaluation of the artefact. If the research incorporated more participants with differing stakeholding interests, that could contribute to a more thorough evaluation. If test case developers or existing testing framework developers provided their perspective of the product, this would enable more feedback on what features could be added or removed.

The visualisation tool that was produced was aimed at a specific test rig at VCC. Future work from this project could investigate the portability of the tool to other testing rigs within VCC and the automotive industry.

Appendix A
Interview questions

1) Is there sufficient information in the test case failures table?
2) What information can be added or removed?
3) Are the bar and pie charts effecting in assisting with fault localisation?
4) Are the categories / sub-categories well defined?
5) When compared to the existing visualisation, how does this impact the fault localisation process?
6) How else could the visualisation be improved?
7) Would you expect the fault localisation process to be improved?
8) Does the page offer an accurate perspective of how the test rig and its test cases are performing?
9) What other information / features could be added to the visualisation?

Appendix B
Interview Notes - Test rig analysis team

- Good to have a **summary** of a **few runs**. **Filter** for a **period range**.
- Add a **link the full html report**. Able to **remove certain runs from statistics**. Exception cases for **setup failures**.
- Good to have, might also be good to use the existing **tags** from **AwesomeFramework**. **Tags** are specified per team and it would be good for teams to be able to focus on their **failures**.
- **Sub-categories** could be better **defined**, for example braking could be distinguished into electrical or parking. Check if some **symbols** have **state machines**.
- Easier in some aspects, but we cant do anything without the existing **AwesomeFramework report**. No need to go through **several week**s of reports to see how **test cases have been failing / succeeding**.
- Could **differentiate** between **errors** that arise from test cases **states** for example. **Delta between different builds** and the **hardware and software versions** associated with the rig clock module revision, **AwesomeFramework** release, has test case been modified.
- Needs **link to test report** itself better **categorisation** but it removes too much **content**. Access to extra logs if its available. **New failures visualisation** is great, unlike before.
- Doesnt show the criticality of the **DTC failures**. Better visualisation of why a test case fails.
- Add information on the system under test. Good if all **tables have filters**.
- More **detail** on being able to see **trends** for each specific test case.
- Possible functionality to be able to add information **(archive) about each test case** in a database.
- Functionality to be able to **compare information between different builds**.
- **Never failed** before is a great feature.
- Maybe **differentiate** why the test has failed.

APPENDIX C
INTERVIEW NOTES - CI TEAM LEADER

- Very good **starting point**, some features need to be **expanded**. The visualisation needs to be constantly evolved to be effective in line with continuous integration.
- More details on the **ECUs**, which areas need to be **investigated**.
- The visualisation improves the **automation** of the **fault localisation**, therefore the whole process has been improved.
- Provides an accurate perspective of how the test rig and its **test cases are performing**.
- Flow issues should be visualised. Test cases could have an **archive of information**, interactive with the user so they can provide and receive information of **previous test case execution**. This provides scope for **machine learning**.
- **Never failed** shows which test cases which need to be investigated as these test cases are dead and may not be finding faults.
- This visualisation contributes a lot to areas which need to be **investigated**.
- The **tags** should be visualised.
- The primary goal of **fault localisation** is to state whether the fault lies in the environment or the software.
- **Statistical analysis** could play a part in the future with improving the **fault localisation** but for now there isnt enough **logging of data**.

ACKNOWLEDGEMENT

The authors would like to thank their supervisor Assistant Professor Dr. *Francisco Gomes de Oliveira Neto* for his guidance and support. Additionally, we would like to thank VCC CI Implementation Leader *Henrik Schreiber* for proposing this topic and allowing this work to be performed at VCC.

REFERENCES

[1] A. Kodali, Y. Zhang, C. Sankavaram, K. Pattipati and M. Salman, *Fault Diagnosis in the Automotive Electric Power Generation and Storage System (EPGS)*, IEEE/ASME Trans.Mechatronics, vol. 18, no. 6, pp. 1809-1817, Dec. 2013.

[2] J. Luo and K. R. Pattipati, *An integrated diagnostic development process for automotive engine control systems*, IEEE Trans. Syst., Man, Cybern.C, vol. 37, no. 6, pp. 11631173, Nov. 2007.

[3] V. N. Malepati, H. Li, K. R. Pattipati, S. Deb, and A. Patterson-Hine, *Verification and validation of high integrity software generated by automatic code generators*, in Proc. IEEE Int. Conf. Syst., Man, Cybern., Oct. 1998, vol. 3, pp. 30043009.

[4] M. Trapp, B. Schurmann, and T. Tetteroo, *Failure behavior analysis for reliable distributed embedded systems*, Int. Parallel Distrib. Process. Symp., pp. 99-107, 2002.

[5] M. Steinder, A. S. Sethi, *A survey of fault localization techniques in computer networks*, Science of Computer Programming, Vol. 53, Issue 2, pp. 165-194, Nov. 2004.

[6] James A. Jones, Mary J. Harrold and John Stasko, *Visualization of Test Information to Assist Fault Localization*, Proceedings of the 24th International Conference on Software Engineering, pp. 467-477, 2002.

[7] L. C. Ascari, L. Y. Araki, A. R. T. Pozo and S. R. Vergilio *Exploring Machine Learning Techniques for Fault Localization*, 10th Latin American Test Workshop, Buzios, Rio de Janeiro, pp. 1-6, 2009.

[8] Guest, Greg; MacQueen, Namey, *Introduction to Thematic Analysis". Applied Thematic Analysis.*, 2012

[9] A. Chunduri, *A survey of fault localization techniques in computer networks*, Science of Computer Programming, Aug. 2016.

[10] L. Mariani, and F. Pastore, *Automated Identification of Failure Causes in System Logs*, 2008.

[11] N. Seth, R. Khare, *ACI (Automated Continuous Integration) using Jenkins: Key for Successful Embedded Software Development*, 2015.

[12] K. R. Pattipati, et al. *An Integrated Diagnostic Process for Automotive Systems*, Oct. 2008.

[13] A. Telea, *Data Visualization: Principles and Practice*, 2nd ed. Boca Ratn, FL: CRC Press, 2014.

[14] B. Kaplan and J. A. Maxwell, *Qualitative Research Methods for Evaluating Computer Information Systems*, Health Informatics. Springer-Verlag, pp. 3055.

[15] Hancock B., Windridge K., and Ockleford E, *An Introduction to Qualitative Research*, The NIHR RDS EM / YH, 2007

[16] Braun, V. and Clarke, V, *Using thematic analysis in psychology. Qualitative Research in Psychology*, 3 (2). pp. 77-101. ISSN 1478-0887, 2006

[17] Richard Boyatzis, *Transforming qualitative information: Thematic analysis and code development*, Thousand Oaks, CA: Sage, 1998

[18] R. Feldt and A. Magazinius, *Validity threats in empirical software engineering research-an initial survey*, in Proceedings of the Conference on Software Engineering and Knowledge Engineering(SEKE), 2010, pp. 374379.

[19] P. Runeson and M. Hst, *Guidelines for conducting and reporting case study research in software engineering*, Empirical Software Engineering, vol. 14, no. 2, pp. 131164, 2008.